FIWARE-PaaS

Release

December 21, 2015

Contents

1 Introduction **1**
1.1 FIWARE PaaS Manager | Pegasus 1
1.2 PaaS Manager - User and Programmers Guide 7
1.3 PaaS Manager - Installation and Administration Guide 19

CHAPTER 1

Introduction

Pegasus is a Java implementation of the PaaS Manager GE developed as a part of the FIWARE platform.

Pegasus orchestrates the provisioning of the required virtual resources at IaaS level and the installation and configuration of the whole software stack of the application, taking into account the underlying virtual infrastructure. It provides a flexible mechanism to perform the deployment, enabling multiple deployment architectures: everything in a single server, several servers, or elastic architectures based on load balancers and different software tiers. Pegasus is a easy way to deploy your applications in the FIWARE Cloud.

The PaaS Manager source code can be found here

This documentation offers deeper information on PaaS Manager.

Documentation

1.1 FIWARE PaaS Manager | Pegasus

- *Introduction*
- *GEi overall description*
- *Why to get it*
- *Build and Install*
  - *Requirements*
  - *Installation*
    - *Using FIWARE package repository (recommended)*
  - *Upgrading from a previous version*
    - *Upgrading database*
    - *Using installation script*
- *Running*
  - *Configuration file*
  - *Checking status*
- *API Overview*
  - *API Reference Documentation*
- *Testing*
  - *Unit tests*
  - *Acceptance tests*
  - *End to End testing*
- *Advanced topics*
- *Support*
- *License*

1.1.1 Introduction

This is the code repository for FIWARE Pegasus, the reference implementation of the PaaS Manager GE.

This project is part of FIWARE. Check also the FIWARE Catalogue - PaaS Manager GE.

Any feedback on this documentation is highly welcome, including bugs, typos or things you think should be included but aren't. You can use FIWARE PaaS Manager - GitHub issues to provide feedback.

For documentation previous to release 4.4.2 please check the manuals at FIWARE public wiki:

- FIWARE PaaS Manager - Installation and Administration Guide
- FIWARE PaaS Manager - User and Programmers Guide

Top

1.1.2 GEi overall description

The PaaS Manager GE provides a new layer over the IaaS layer (Openstack) in the aim of easing the task of deploying applications on a Cloud infrastructure. Therefore, it orchestrates the provisioning of the required virtual resources at IaaS level, and then, the installation and configuration of the whole software stack of the application by the SDC GE ((see FIWARE SDC), taking into account the underlying virtual infrastructure. It provides a flexible mechanism to perform the deployment, enabling multiple deployment architectures: everything in a single VM or server, several VMs or servers, or elastic architectures based on load balancers and different software tiers.

Top

1.1.3 Why to get it

PaaS Manager GE is the orchestration platform to be used in the FIWARE Cloud ecosystem to deploy not just insfrastructure but also software on top of that.

- **Full Openstack integrated solution** The PaaS Manager is fully integrated with the Opesntack services (nova for computation, neutron for networking and glance for image catalog.
- **Asynchronous interface**

  Asynchronous interface with polling mechanism to obtain information about the deployment status.
- **Decoupling the management and provisioning**

  Decoupling the management of the catalogue (specifications of what can be deployed) and the management of the inventory (instances of what has been already deployed). In addition, decoupling the management of environments from the management of applications, since there could be uses cases where the users of those functionalities could be different ones.

Top

1.1.4 Build and Install

The recommended procedure is to install using RPM packages in CentOS 6.x as it is explained in the following document . If you are interested in building from sources, check this document.

Requirements

- System resources: see these recommendations.
- Operating systems: CentOS (RedHat), being CentOS 6.5 the reference operating system.

Installation

Using FIWARE package repository (recommended)

Refer to the documentation of your Linux distribution to set up the URL of the repository where FIWARE packages are available (and update cache, if needed):

```
http://repositories.testbed.fiware.org/repo/rpm/x86_64
```

Then, use the proper tool to install the packages:

```
$ sudo yum install fiware-paas
```

and the latest version will be installed. In order to install a specific version:

```
$ sudo yum install fiware-paas-{version}-1.noarch
```

where {version} being the specific version to be installed

Upgrading from a previous version

Unless explicitly stated, no migration steps are required to upgrade to a newer version of the PaaS Manager components:

- When using the package repositories, just follow the same directions described in the *Installation* section (the `install` subcommand also performs upgrades).
- When upgrading from downloaded package files, use `rpm -U` in CentOS

Upgrading database

In case the database needs to be upgrade, the script db-changelog.sql should be execute. To do that, it just needed to execute:

```
psql -U postgres -d $db_name << EOF
\i db-changelog.sql
```

Using installation script

The installation of fiware-paas can be done in the easiest way by executing the script:

```
scripts/bootstrap/centos.sh
```

The script will ask you the following data to configure the configuration properties:

- The database name for the fiware-paas
- The postgres password of the database
- the keystone url to connect fiware-paas for the authentication process
- the admin keystone user for the authentication process
- the admin password for the authentication process

Top

1.1.5 Running

As explained in the *GEi overall description* section, there are a variety of elements involved in the PaaS Manager architecture, apart from those components provided by this PaaS Manager GE as the Software Deployment and Configuration and OpenStack services. Please refer to their respective documentation for instructions to run them.

In order to start the PaaS Manager service, as it is based on a web application on top of jetty, just you should run:

```
$ service fiware-paas start
```

Then, to stop the service, run:

```
$ service fiware-paas stop
```

We can also force a service restart:

```
$ service fiware-paas restart
```

Configuration file

The configuration of PaaS Manager is in configuration_properties table in the database. There, it is required to configure:

```
$ openstack-tcloud.keystone.url: This is the url where the keystone-proxy is deployed
$ openstack-tcloud.keystone.user: the admin user
$ openstack-tcloud.keystone.password: the admin password
$ openstack-tcloud.keystone.tenant: the admin tenant
$ paas_manager_url: the final url, mainly https://paas-ip:8443/paasmanager
```

In addition, to configue the PaaS Manager application inside the webserver, it is needed to change the context file. To do that, change paasmanager.xml found in distribution file and store it in folder $PAASMANAGER_HOME/webapps/:

```
<New id="sdc" class="org.eclipse.jetty.plus.jndi.Resource">
  <Arg>jdbc/paasmanager</Arg>
  <Arg>
      <New class="org.postgresql.ds.PGSimpleDataSource">
          <Set name="User"> <database user> </Set>
          <Set name="Password"> <database password> </Set>
          <Set name="DatabaseName"> <database name>   </Set>
          <Set name="ServerName"> <IP/hostname> </Set>
          <Set name="PortNumber">5432</Set>
      </New>
  </Arg>
</New>
```

Checking status

In order to check the status of the service, use the following command (no special privileges required):

```
$ service fiware-paas status
```

Top

1.1.6 API Overview

The PaaS Manager offers a REST API, which can be used for both managing deploying virtual infrastructure and install software on top of it.

For instance, it is possible to obtain the template list in the catalogue:

```
$ curl -v -H "Content-Type: application/json" -H "Accept: application/xml" -H "X-Auth-Token: your-to
  -X GET "https://pegasus.lab.fi-ware.org:8443/paasmanager/rest/catalog/org/FIWARE/environment"
```

Please have a look at the API Reference Documentation section bellow and at the programmer guide.

API Reference Documentation

- FIWARE PaaS Manager v1 (Apiary)

Top

1.1.7 Testing

Unit tests

The `test` target for each module in the PaaS Manager is used for running the unit tests in both components of PaaS Manager GE. To execute the unit tests you just need to execute:

```
mvn test
```

Please have a look at the section building from source code in order to get more information about how to prepare the environment to run the unit tests.

Acceptance tests

In the following path you will find a set of tests related to the end-to-end funtionalities.

- PaaS Manager Aceptance Tests

To execute the acceptance tests, go to the test/acceptance folder of the project and run:

```
lettuce_tools --tags=-skip.
```

This command will execute all acceptance tests (see available params with the -h option)

End to End testing

Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. It involves to obtain the product information storaged in the catalogue. With it, we test that the service is running and the database configure correctly:

```
https://{PaaS Manager\_IP}:{port}/paasmanager/rest
```

The request to test it in the testbed should be:

```
curl -v -k -H 'Access-Control-Request-Method: GET' -H 'Content-Type: application xml' -H 'Accept: app
-H 'X-Auth-Token: 5d035c3a29be41e0b7007383bdbbec57' -H 'Tenant-Id: 60b4125450fc4a109f50357894ba2e28'
-X GET 'https://localhost:8443/paasmanager/rest/catalog/org/FIWARE/environment'
```

the option -k should be included in the case you have not changed the security configuration of PaaS Manager. The result should be the product catalog.

If you obtain a 401 as a response, please check the admin credentials and the connectivity from the PaaS Manager machine to the keystone (openstack-tcloud.keystone.url in configuration_properties table)

Top

1.1.8 Advanced topics

- Installation and administration
  - Software requirements
  - Building from sources
  - Resources & I/O Flows
- User and programmers guide

Top

1.1.9 Support

Ask your thorough programming questions using stackoverflow and your general questions on FIWARE Q&A. In both cases please use the tag *fiware-pegasus*

Top

1.1.10 License

(c) 2013-2015 Telefónica I+D, Apache License 2.0

Top

1.2 PaaS Manager - User and Programmers Guide

1.2.1 Introduction

Welcome the User and Programmer Guide for the PaaS Manager GE. This generic enabler is built on a proprietary solution using standard interface to communicate with and so where possible this guide points to the appropriate online content that has been created for this specific API. The online documents are being continuously updated and improved, and so will be the most appropriate place to get the most up to date information on using this interface.

1.2.2 Accessing PaaS Manager from the CLI

The access through the CLI is made using the curl program. Curl [http://curl.haxx.se/] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, GOPHER, DICT, TELNET, LDAP or FILE) and therefore is also usable for OpenStack Compute API. Use the curl command line tool or use libcurl from within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

The normal operations sequence to deploying an environment and an application on top of it could be summarized in the following list:

API Authentication

All the operations in the PaaS Manager API needs to have a valid token to access it. To obtain the token, you need to have an account in FIWARE Lab (account.lab.fi-ware.org). With the credentials (username, password and tenantName) you can obtain a valid token. From now on, we asume that the value of your tenant-id is "your-tenant-id"

```
$ curl -v -H "Content-Type: application/json" -H "Accept: application/json" -X
POST "http://cloud.lab.fi-ware.org:4731/v2.0/tokens" -d '{"auth":{"tenantName":
"your-tenant-id","passwordCredentials":{"username":"youruser","password":"yourpassword"}}}'
```

You will receive the following answer, with a valid token (id).

```
{
access: {
   token: {
      expires: "2015-07-09T15:16:07Z"
      id: "756cfb31e062216544215f54447e2716"
      tenant: {
      ..
}
```

For all the PaaS manager request, you will need to include the following header:

```
X-Auth-Token: 756cfb31e062216544215f54447e2716
Tenant-Id: your-tenant-id
```

For the rest of the explanation, we are going to configure a set of variables:

```
export PAAS_MANAGER_IP =  pegasus.lab.fi-ware.org
```

Abstract Environment API

Next we detail some operations that can be done in the catalogue managemente api regarding the Abstract Environments. Abstract Environments are environments defined by the administrator. They are available for all FIWARE users.

Get the Abstract Environment list from the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment"
```

This operation lists the abstract environments stored in the catalogue. The following example shows an XML response for the list Abstract Environment API operation.

```
 <environmentDtoes>
     <environmentDto>
                    <tierDtos>
                            <name>orion</name>
                            <flavour>2</flavour>
                            <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
                            <maximumNumberInstances>1</maximumNumberInstances>
                            <minimumNumberInstances>1</minimumNumberInstances>
                            <initialNumberInstances>1</initialNumberInstances>
                            <productReleaseDtos>
                                    <productName>orion</productName>
                                    <version>0.13.0</version>
                            </productReleaseDtos>
                            <icono />
                            <securityGroup />
                            <keypair />
                            <floatingip>false</floatingip>
                            <affinity>None</affinity>
                            <region>Spain</region>
                    </tierDtos>
                    <name>orion</name>
                    <description>Environment orion</description>
             </environmentDto>
             ...
</environmentDtoes>
```

Get a particular Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-er
```

This operation lists the abstract environments stored in the catalogue. The following example shows an XML response for the list Abstract Environment API operation.

```
 <environmentDtoes>
     <environmentDto>
                     <tierDtos>
                             <name>{abstract-environment-name}</name>
                             <flavour>2</flavour>
                             <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
                             <maximumNumberInstances>1</maximumNumberInstances>
                             <minimumNumberInstances>1</minimumNumberInstances>
                             <initialNumberInstances>1</initialNumberInstances>
                             <productReleaseDtos>
                                     <productName>orion</productName>
                                     <version>0.13.0</version>
                             </productReleaseDtos>
                             <icono />
                             <securityGroup />
                             <keypair />
                             <floatingip>false</floatingip>
                             <affinity>None</affinity>
                             <region>Spain</region>
                     </tierDtos>
                     <name>orion</name>
                     <description>Environment orion</description>
             </environmentDto>
</environmentDtoes>
```

Add an Abstract Environment to the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X POST "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment"
```

with the following payload

```
<?xml version="1.0" encoding="UTF-8"?>
<environmentDto>
    <name>{abstract-environment-name}</name>
    <description>description</description>
</environmentDto>
```

Delete an abstract template for the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X DELETE "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract
```

Abstract Tier API

Add an Tier to an existing Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X POST "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-e
```

with the following payload

```
<tierDto>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
```

```
        <maximumNumberInstances>1</maximumNumberInstances>
        <name>{tier-name}</name>
        <image>0dbf8aff-5dc5-4d6c-9f9c-1e6801e0b629</image>
        <flavour>2</flavour>
        <keypair>jesusmmovilla57</keypair>
        <floatingip>false</floatingip>
        <region>Trento</region>
</tierDto>
```

Get All Tiers associated to a Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-er
```

This operation obtains a response with the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<tierDtoes>
        <tierDto>
                <name>{tier-name}</name>
                <flavour>2</flavour>
                <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
                <maximumNumberInstances>3</maximumNumberInstances>
                <minimumNumberInstances>1</minimumNumberInstances>
                <initialNumberInstances>1</initialNumberInstances>
                <productReleaseDtos>
                        <productName>mongodbshard</productName>
                        <productDescription>mongodb shard 2.2.3</productDescription>
                        <version>2.2.3</version>
                </productReleaseDtos>
                <icono>http://blog.theinit.com/wp-content/uploads/2012/03/bc358_MongoDB.png</icono>
                <securityGroup />
                <keypair />
                <floatingip>false</floatingip>
                <affinity>None</affinity>
                <region>Spain</region>
        </tierDto>
</tierDtoes>
```

Get a particular Tier associated to a Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-er
```

This operation obtains a response with the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<tierDto>
        <name>{tier-name}</name>
        <flavour>2</flavour>
        <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
        <maximumNumberInstances>3</maximumNumberInstances>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <productReleaseDtos>
                <productName>mongodbshard</productName>
                <productDescription>mongodb shard 2.2.3</productDescription>
                <version>2.2.3</version>
```

```
        </productReleaseDtos>
        <icono>http://blog.theinit.com/wp-content/uploads/2012/03/bc358_MongoDB.png</icono>
        <securityGroup />
        <keypair />
        <floatingip>false</floatingip>
        <affinity>None</affinity>
        <region>Spain</region>
</tierDto>
```

Update a Tier of an existing Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X PUT "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-en
```

with the following payload

```
<tierDto>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <maximumNumberInstances>1</maximumNumberInstances>
        <name>{tier-name}</name>
        <image>0dbf8aff-5dc5-4d6c-9f9c-1e6801e0b629</image>
        <flavour>2</flavour>
        <keypair>jesusmmovilla57</keypair>
        <floatingip>false</floatingip>
        <region>Spain</region>
</tierDto>
```

Delete a particular Tier associated to a Abstract Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/environment/{abstract-en
```

Blueprint Template/Environment API

Next we detail some operations that can be done in the catalogue managemente api

Get the blueprint template list from the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/env
```

This operation lists the environments stored in the catalogue. The following example shows an XML response for the list Environment API operation. It is possible to see it contains a list of tiers including products to be installed.

```
 <environmentDtoes>
     <environment>
         <name>{emvironment-name}</name>
         <tiers>
             <tier>
                 <initial_number_instances>1</initial_number_instances>
                 <maximum_number_instances>1</maximum_number_instances>
                 <minimum_number_instances>1</minimum_number_instances>
                 <name>{tier-id}</name>
                 <networkDto>
```

```
                    <networkName>Internet</networkName>
                    <subNetworkDto>
                            <subnetName>sub-net-Internet</subnetName>
                    </subNetworkDto>
                </networkDto>
                <productReleases>
                    <product>postgresql</product>
                    <version>0.0.3</version>
                    <withArtifact>true</withArtifact>
                    <productType>
                        <id>5</id>
                        <name>Database</name>
                    </productType>
                </productReleases>
                ...
            </tier>
        </tiers>
    </environment>
    <environment>
        <name>{emvironment-name}</name>
        <tiers>
            <tier>
            ...
            </tier>
        </tiers>
    </environment>
</environmentDtoes>
```

Add a blueprint template to the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X POST "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/en
```

with the following payload

```
<?xml version="1.0" encoding="UTF-8"?>
<environmentDto>
    <name>{environment-name}</name>
    <description>{description of environment}</description>
    <tierDtos>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <maximumNumberInstances>1</maximumNumberInstances>
        <name>{tier-name}</name>
        <networkDto>
            <networkName>{network-name}</networkName>
                <subNetworkDto>
                        <subnetName>{subnetwork-name}</subnetName>
                </subNetworkDto>
        </networkDto>
        <image>{image-id}</image>
        <flavour>{flavour of VM in number}</flavour>
        <keypair>{keypair-name}</keypair>
        <floatingip>{false/true}</floatingip>
        <region>{region-name}</region>
        <productReleaseDtos>
                <productName>{product-name}</productName>
                <version>{product-version}</version>
```

```
        </productReleaseDtos>
    </tierDtos>
</environmentDto>
```

The network and region information are including also in the payload of the environment. The following lines show a example.

```
<tierDtos>
    ...
    <name>{tier-name}</name>
        <networkDto>
            <networkName>{network-name}</networkName>
                <subNetworkDto>
                        <subnetName>{subnetwork-name}</subnetName>
                </subNetworkDto>
        </networkDto>
        <image>{image-id}</image>
        <flavour>{flavour of VM in number}</flavour>
        <keypair>{keypair-name}</keypair>
        <floatingip>{false/true}</floatingip>
        <region>{region-name}</region>
        <productReleaseDtos>
                <productName>{product-name}</productName>
                <version>{product-version}</version>
        </productReleaseDtos>
        ...
</tierDtos>
```

Delete a blueprint template from the catalogue

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X DELETE "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/
```

Tier API

Add a Tier to an existing Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X POST "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/en
```

with the following payload

```
<tierDto>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <maximumNumberInstances>1</maximumNumberInstances>
        <networkDto>
                <networkName>Internet</networkName>
                <subNetworkDto>
                        <subnetName>sub-net-Internet</subnetName>
                </subNetworkDto>
        </networkDto>
        <name>{tier-name}</name>
        <image>0dbf8aff-5dc5-4d6c-9f9c-1e6801e0b629</image>
        <flavour>2</flavour>
        <keypair>jesusmmovilla57</keypair>
```

```
        <floatingip>false</floatingip>
        <region>Trento</region>
</tierDto>
```

Get All Tiers associated to an Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/env
```

This operation obtains a response with the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<tierDtoes>
        <tierDto>
                <name>{tier-name}</name>
                <flavour>2</flavour>
                <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
                <maximumNumberInstances>3</maximumNumberInstances>
                <minimumNumberInstances>1</minimumNumberInstances>
                <initialNumberInstances>1</initialNumberInstances>
                <networkDto>
                        <networkName>Internet</networkName>
                        <subNetworkDto>
                                <subnetName>sub-net-Internet</subnetName>
                        </subNetworkDto>
                </networkDto>
                <productReleaseDtos>
                        <productName>mongodbshard</productName>
                        <productDescription>mongodb shard 2.2.3</productDescription>
                        <version>2.2.3</version>
                </productReleaseDtos>
                <icono>http://blog.theinit.com/wp-content/uploads/2012/03/bc358_MongoDB.png</icono>
                <securityGroup />
                <keypair />
                <floatingip>false</floatingip>
                <affinity>None</affinity>
                <region>Spain</region>
        </tierDto>
</tierDtoes>
```

Get a particular Tier associated to an Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/env
```

This operation obtains a response with the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<tierDto>
        <name>{tier-name}</name>
        <flavour>2</flavour>
        <image>dbefb2d6-2221-46e2-a11c-b466e2503da5</image>
        <maximumNumberInstances>3</maximumNumberInstances>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <networkDto>
                <networkName>Internet</networkName>
                <subNetworkDto>
```

```
                        <subnetName>sub-net-Internet</subnetName>
                </subNetworkDto>
        </networkDto>
        <productReleaseDtos>
                <productName>mongodbshard</productName>
                <productDescription>mongodb shard 2.2.3</productDescription>
                <version>2.2.3</version>
        </productReleaseDtos>
        <icono>http://blog.theinit.com/wp-content/uploads/2012/03/bc358_MongoDB.png</icono>
        <securityGroup />
        <keypair />
        <floatingip>false</floatingip>
        <affinity>None</affinity>
        <region>Spain</region>
</tierDto>
```

Update a Tier of an existing Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X PUT "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/env
```

with the following payload

```
<tierDto>
        <minimumNumberInstances>1</minimumNumberInstances>
        <initialNumberInstances>1</initialNumberInstances>
        <maximumNumberInstances>1</maximumNumberInstances>
        <name>{tier-name}</name>
        <networkDto>
                <networkName>Internet</networkName>
                <subNetworkDto>
                        <subnetName>sub-net-Internet</subnetName>
                </subNetworkDto>
        </networkDto>
        <image>0dbf8aff-5dc5-4d6c-9f9c-1e6801e0b629</image>
        <flavour>2</flavour>
        <keypair>jesusmmovilla57</keypair>
        <floatingip>false</floatingip>
        <region>Spain</region>
</tierDto>
```

Delete a particular Tier associated to an Environment

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/{your-tenant-id}/env
```

BluePrint/Environment Instance Provisioning API

Deploy a Blueprint Instance

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
-X POST "https://PAAS_MANAGER_IP:8443/paasmanager/rest/envInst/org/FIWARE/vdc/{your-tenant-id}/enviro
```

where “your-tenant-id” is the tenant-id in this guide. The payload of this request can be as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environmentInstanceDto>
        <blueprintName>{environmentinstance-name}</blueprintName>
        <description>{description of environmentinstance}</description>
        <environmentDto>
                <name>{environment-name}</name>
                <description>{description of environmet}</description>
                <tierDtos>
                        <name>{tier-name}</name>
                        <flavour>{flavour of the VM}</flavour>
                        <image>{image-id of the image to create the VM}</image>
                        <maximumNumberInstances>1</maximumNumberInstances>
                        <minimumNumberInstances>1</minimumNumberInstances>
                        <initialNumberInstances>1</initialNumberInstances>
                        <networkDto>
                                <networkName>{network-name}</networkName>
                        </networkDto>
                        <icono></icono>
                        <securityGroup>{security-group-name}</securityGroup>
                        <keypair>{keypair-name}</keypair>
                        <floatingip>{true/false}</floatingip>
                        <affinity>None</affinity>
                        <region>{region-name where to deploy}</region>
                </tierDtos>
        </environmentDto>
</environmentInstanceDto>
```

The response obatined should be:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/your-tenant-id/task/
    <description>Deploy environment {emvironment-name}</description>
    <vdc>your-tenant-id</vdc>
</task>
```

Given the URL obtained in the href in the Task, it is possible to monitor the operation status (you can check Task Management). Once the environment has been deployed, the task status should be SUCCESS.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="https://PAAS_MANAGER_IP:8443/paasmanager/rest/catalog/org/FIWARE/vdc/your-tenant-id/task/
    <description>Deploy environment {emvironment-name}</description>
    <vdc>your-tenant-id</vdc>
</task>
```

Get information about Blueprint Instances deployed

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
-X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/envInst/org/FIWARE/vdc/your-tenant-id/environme
```

The Response obtained includes all the blueprint instances deployed

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <environmentInstanceDtoes>
     <environmentInstance>
         <environmentInstanceName>{environmentInstance-id</environmentInstanceName>
         <vdc>your-tenant-id</vdc>
         <environment>
             <name>{environment-name}</name>
             <tiers>
```

```
                <tier>
                <initial_number_instances>1</initial_number_instances>
                <maximum_number_instances>1</maximum_number_instances>
                <minimum_number_instances>1</minimum_number_instances>
                <name>{tier-id}</name>
                <productReleases>
                    <product>postgresql</product>
                    <version>0.0.3</version>
                    <withArtifact>true</withArtifact>
                    <productType>
                        <id>5</id>
                        <name>Database</name>
                    </productType>
                </productReleases>                      ...
                </tier>
            </tiers>
        </environment>
        <tierInstances>
            <id>35</id>
            <date>2012-10-31T09:24:45.298Z</date>
            <name>tomcat-</name>
            <status>INSTALLED</status>
            <vdc>your-tenant-id</vdc>
            <tier>
                <name>{tier-id}</name>
            </tier>
            <productInstances>
                <id>33</id>
                <date>2012-10-31T09:14:33.192Z</date>
                <name>postgresql</name>
                <status>INSTALLED</status>
                <vdc>your-tenant-id</vdc>
                <productRelease>
                    <product>postgresql</product>
                    <version>0.0.3</version>
                </productRelase>
                <vm>
                    <fqn>vmfqn</fqn>
                    <hostname>rehos456544</hostname>
                    <ip>109.231.70.77</ip>
                </vm>
        </tierInstances>
    </environmentInstance>
</environmentInstanceDtoes>
```

Get details of a certain Blueprint Instance

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
-X GET "https://PAAS_MANAGER_IP:8443/paasmanager/rest/envInst/org/FIWARE/vdc/your-tenant-id/environme
```

This operation does not require any payload in the request and provides a BlueprintInstance XML response.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environmentInstancePDto>
            <environmentInstanceName>{environmentinstance-name}</environmentInstanceName>
            <vdc>{tenant-id}</vdc>
            <description>{description of environmentinstance}</description>
            <status>{status of the environment installation}</status>
```

```
            <blueprintName>{blueprint-name}</blueprintName>
            <taskId>{task-id of the execution}</taskId>
            <tierDto>
                    <name>{tier-name}</name>
                    <flavour>{flavour of the vm}</flavour>
                    <image>{image-id}</image>
                    <maximumNumberInstances>1</maximumNumberInstances>
                    <minimumNumberInstances>1</minimumNumberInstances>
                    <initialNumberInstances>1</initialNumberInstances>
                    <productReleaseDtos>
                            <productName>{product-name}</productName>
                            <version>{product-version}</version>
                    </productReleaseDtos>
                    <icono />
                    <securityGroup>{securityGroup-name}</securityGroup>
                    <keypair>{keypair-name}</keypair>
                    <floatingip>{true/false}</floatingip>
                    <region>{region-name}</region>
                    <tierInstancePDto>
                            <tierInstanceName>{tierinstance-name}</tierInstanceName>
                            <status>{status of the tierinstallation}</status>
                            <taskId>{task id of tier installation execution}</taskId>
                            <productInstanceDtos>
                                    <productReleaseDto>
                                            <productName>{product-name}</productName>
                                            <version>{product-version}</version>
                                    </productReleaseDto>
                                    <name>{productInstance-name}</name>
                                    <taskId>{task id of product installation}</taskId>
                            </productInstanceDtos>
                            <vm>
                                    <domain>{domain of vm}</domain>
                                    <fqn>{fqn of vm}</fqn>
                                    <hostname>{hostname}</hostname>
                                    <ip>{ip}</ip>
                                    <id>{nova-host-id}</id>
                            </vm>
                    </tierInstancePDto>
            </tierDto>
    </environmentInstancePDto>
```

Undeploy a Blueprint Instance

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
-X DELETE "https://PAAS_MANAGER_IP:8443/paasmanager/rest/envInst/org/FIWARE/vdc/{your-tenant-id}/envi
```

This operation does not require a request body and returns the details of a generated task.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="https://PAAS_MANAGER_IP:8443/paasmanager/rest/vdc/{your-tenant-id}/task/{task-id}" startT
    <description>Uninstall environment</description>
    <vdc>your-tenant-id</vdc>
</task>
```

With the URL obtained in the href in the Task, it is possible to monitor the operation status (you can checkTask Management). Once the environment has been undeployed, the task status should be SUCCESS.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="https://PAAS_MANAGER_IP:8443/paasmanager/rest/vdc/{your-tenant-id}/task/{task-id}" start
    <description>Undeploy environment {emvironment-name}</description>
    <vdc>your-tenant-id</vdc>
</task>
```

Task Management

Get a specific task

```
$ curl -v -H "Content-Type: application/xml" -H "Accept: application/xml" -H
"X-Auth-Token: 756cfb31e062216544215f54447e2716" -H "Tenant-Id: your-tenant-id"
    -X DELETE "http://pegasus.lab.fi-ware.org:8080/paasmanager/rest/vdc/your-tenant-id/task/{task-id
```

This operation recovers the status of a task created previously. It does not need any request body and the response body in XML would be the following.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <task href="http:/130.206.80.112:8080/sdc/rest/vdc/{your-tenant-id}/task/{task-id}" startTime="20
    <description>Install product tomcat in  VM rhel-5200ee66c6</description>
    <vdc>your-tenant-id</vdc>
</task>
```

The value of the status attribute could be one of the following:

Value	Description
QUEUED	The task is queued for execution.
PENDING	The task is pending for approval.
RUNNING	The task is currently running.
SUCCESS	The task is completed successfully.
ERROR	The task is finished but it failed.
CANCELLED	The task has been cancelled by user.

1.3 PaaS Manager - Installation and Administration Guide

1.3.1 Introduction

This guide defines the procedure to install the different components that build up the PaaS Manager GE, including its requirements and possible troubleshooting. The guide includes two different ways of installing PaaS Manager: Installation from rpm or installation from source (building previously the rpm).

1.3.2 Requirements

In order to execute the PaaS Manager, it is needed to have previously installed the following software:

- PostgreSQL.

You can find a small guide to install PostgresSQL in the next section. If you find some problems installing PostgreSQL, please refer to the postgres official site.

PaaS Manager should be installed in a host with 2Gb RAM.

1.3.3 Installation from script

The installation of fiware-paas can be done in the easiest way by executing the script

```
scripts/bootstrap/centos.sh
```

that is in the github repository of the project.

In order to perform the installation via script, git should be installed (yum install git). Just clone the github repository:

```
git clone https://github.com/telefonicaid/fiware-paas
```

and go to the folder

```
cd fiware-paas/scripts/bootstrap
```

Assign the corresponding permissions to the script centos.sh and execute under root user

```
./centos.sh
```

The script will ask you the following data:

- The database name for the fiware-paas
- The postgres password of the database
- the keytone url to connect fiware-paas for the uthentication process
- the admin keystone user for the autentication process
- the admin password for the autentication process

Once the script is finished, you will have fiware-paas installed under /opt/fiware-paas/ . Please go to the Sanity Check section in order to test the installation. This script does not insert the fiware-paas data into the keystone, so this action has to be done manually. In order to complete the installation please refer to Configuring the PaaS Manager in the kesytone section.

1.3.4 Manual Installation

Requirements: Install PostgreSQL

The first thing is to install and configure the requirements, in this case, the postgresql

```
    yum install postgresql postgresql-server postgresql-contrib
```

Type the following commands to install the postgresql as service and start it

```
chkconfig --add postgresql
chkconfig postgresql on
service postgresql initdb
service postgresql start
```

Install PaaS Manager from RPM

The PaaS Manager is packaged as RPM and stored in the rpm repository. Thus, the first thing to do is to create a file in /etc/yum.repos.d/fiware.repo, with the following content.

```
[Fiware]
name=FIWARE repository
baseurl=http://repositories.testbed.fi-ware.eu/repo/rpm/x86_64/
gpgcheck=0
enabled=1
```

After that, you can install the PaaS Manager just doing:

```
yum install fiware-paas
```

or specifying the version

```
yum install fiware-paas-{version}-1.noarch
```

where {version} could 1.5.0

Install PaaS Manager from source

Requirements: To install Paas Manager from source it is required to have the following software installed in your host previously:

- git
- java 1.7
- maven

Here we include a small guide to install the required software. If you find any problem in the installation process, please refer to the official site:

Install git

```
sudo yum install git
```

Install java 1.7

```
sudo yum install java-1.7.0-openjdk-devel
```

Install maven 2.5

```
sudo yum install wget
wget http://mirrors.gigenet.com/apache/maven/maven-3/3.2.5/binaries/apache-maven-3.2.5-bin.tar.gz
su -c "tar -zxvf apache-maven-3.2.5-bin.tar.gz -C /usr/local"
cd /usr/local
sudo ln -s apache-maven-3.2.5 maven
```

Add the following lines to the file /etc/profile.d/maven.sh

```
# Add the following lines to maven.sh
export M2_HOME=/usr/local/maven
export M2=$M2_HOME/bin
PATH=$M2:$PATH
```

In order to check that your maven installation is OK, you shluld exit your current session with “exit” command, enter again and type

```
mvn -version
```

if the system shows the current maven version installed in your host, you are ready to continue with this guide.

Now we are ready to build the PaaS Manager rpm and finally install it

The PaaS Manager is a maven application so, we should follow following instructions:

- Download PaaS Manager code from github

```
git clone -b develop https://github.com/telefonicaid/fiware-paas
```

- Go to fiware-paas folder and compile, launch test and build all modules

```
cd fiware-paas/
mvn clean install
```

- Create a zip with distribution in target/paas-manager-server-dist.zip

```
mvn assembly:assembly -DskipTests
```

- You can generate a rpm o debian packages (using profiles in pom) for debian/ubuntu:

```
mvn install -Pdebian -DskipTests
     (created target/paas-manager-server-XXXXX.deb)
```

- for centOS (you need to have installed rpm-bluid. If not, please type "yum install rpm-build")

```
mvn install -Prpm -DskipTests
    (created target/rpm/paasmanager/RPMS/noarch/paasmanager-XXXX.noarch.rpm)
```

Finally go to the folder where the rpm has been created (target/rpm/fiware-paas/RPMS/noarch) and execute

```
cd target/rpm/fiware-paas/RPMS/noarch
rpm -i <rpm-name>.rpm
```

Please, be aware that the supported installation method is the RPM package. If you use other method, some extra steps may be required. For example you would need to generate manually the certificate (See the section about "Configuring the HTTPS certificate" for more information):

```
fiware-paas/bin/generateselfsigned.sh
```

Configuring the database

We need to create the paasmanager database. To do that we need to connect as postgres user to the PostgreSQL server and set the password for user postgres using alter user as below:

```
su - postgres
postgres$ psql postgres postgres;
psql (8.4.13)
Type "help" for help.
postgres=# alter user postgres with password 'postgres';
postgres=# create database paasmanager;
postgres=# grant all privileges on database paasmanager to postgres;
postgres=#\q
exit
```

Edit file /var/lib/pgsql/data/pg_hba.conf and set authentication method to md5:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
  "local" is for Unix domain socket connections only
  local   all         all                               md5
  local   all         postgres                          md5
# IPv4 local connections:
  host    all         all         0.0.0.0/0             md5
```

Edit file /var/lib/pgsql/data/postgresql.conf and set listen addresses to 0.0.0.0:

```
listen_addresses = '0.0.0.0'
```

Reload configuration

```
service postgresql reload
```

To create the tables in the databases, just go to

```
su - potgres
cd /opt/fiware-paas/resources
postgres$ psql -U postgres -d paasmanager
Password for user postgres: <postgres-password-previously-chosen>
postgres=# \i db-initial.sql
postgres=# \i db-changelog.sql
exit
```

Update the following columns in the table configuration_properties:

```
openstack-tcloud.keystone.url=<keystone.url>
paas_manager_url=https://{ip}:8443/paasmanager/rest
openstack-tcloud.keystone.user= <keystone.user>
openstack-tcloud.keystone.pass= <keystone.password>
openstack-tcloud.keystone.tenant=<keystone.tenant>
user_data_path=/opt/fiware-paas/resources/userdata
```

where the values between bracket <> should be found out depending on the openstack installation. The updates of the columns are done in the following way

```
su - potgres
postgres$ psql -U postgres -d paasmanager
Password for user postgres: <postgres-password-previously-chosen>
postgres=# UPDATE configuration_properties SET value='/opt/fiware-paas/resources/userdata' where key=
postgres=# UPDATE configuration_properties SET value='<the value>' where key='paas_manager_url';
postgres=# UPDATE configuration_properties SET value='<the value>' where key='openstack-tcloud.keysto
postgres=# UPDATE configuration_properties SET value='<the value>' where key='openstack-tcloud.keysto
postgres=# UPDATE configuration_properties SET value='<the value>' where key='openstack-tcloud.keysto
```

Configure PaaS Manager application

Once the prerequisites are satisfied, you shall modify the context file at /opt/fiware-paas/webapps/paasmanager.xml

See the snipet bellow to know how it works:

```
<New id="paasmanager" class="org.eclipse.jetty.plus.jndi.Resource">
   <Arg>jdbc/paasmanager</Arg>
   <Arg>
       <New class="org.postgresql.ds.PGSimpleDataSource">
           <Set name="User"> {database user} </Set>
           <Set name="Password"> {database password} </Set>
           <Set name="DatabaseName"> {database name}   </Set>
           <Set name="ServerName"> {IP database hostname - localhost default} </Set>
           <Set name="PortNumber"> {port database - 5432 default}</Set>
       </New>

   </Arg>
</New>
```

Configuring the PaaS Manager as service

Once we have installed and configured the PaaS Manager, the next step is to configure it as a service. To do that just create a file in /etc/init.d/fiware-paas with the following content

```
#!/bin/bash
# chkconfig: 2345 20 80
# description: Description comes here....
# Source function library.
. /etc/init.d/functions
start() {
    /opt/fiware-paas/bin/jetty.sh start
}
stop() {
    /opt/fiware-paas/bin/jetty.sh stop
}
case "$1" in
    start)
        start
    ;;
    stop)
        stop
    ;;
    restart)
        stop
        start
    ;;
    status)
        /opt/fiware-paas/bin/jetty.sh status
    ;;
    *)
        echo "Usage: $0 {start|stop|status|restart}"
esac
exit 0
```

Now you need to execute:

```
chkconfig --add fiware-paas
chkconfig fiware-paas on
service fiware-paas start
```

Configuring the HTTPS certificate

The service is configured to use HTTPS to secure the communication between clients and the server. One central point in HTTPS security is the certificate which guarantee the server identity.

Quickest solution: using a self-signed certificate

The service works "out of the box" against passive attacks (e.g. a sniffer) because a self-signed certificated is generated automatically when the RPM is installed. Any certificate includes a special field call "CN" (Common name) with the identity of the host: the generated certificate uses as identity the IP of the host.

The IP used in the certificate should be the public IP (i.e. the floating IP). The script which generates the certificate knows the public IP asking to an Internet service (http://ifconfig.me/ip). Usually this obtains the floating IP of the server, but of course it wont work without a direct connection to Internet.

If you need to regenerate a self-signed certificate with a different IP address (or better, a convenient configured hostname), please run:

```
/opt/fiware-paas/bin/generateselfsigned.sh myhost.mydomain.org
```

By the way, the self-signed certificate is at /etc/keystorejetty. This file wont be overwritten although you reinstall the package. The same way, it wont be removed automatically if you uninstall de package.

Advanced solution: using certificates signed by a CA

Although a self-signed certificate works against passive attack, it is not enough by itself to prevent active attacks, specifically a "man in the middle attack" where an attacker try to impersonate the server. Indeed, any browser warns user against self-signed certificates. To avoid these problems, a certificate conveniently signed by a CA may be used.

If you need a certificate signed by a CA, the more cost effective and less intrusive practice when an organization has several services is to use a wildcard certificate, that is, a common certificate among all the servers of a DNS domain. Instead of using an IP or hostname in the CN, an expression as ".fiware.org " is used.

This solution implies:

- The service must have a DNS name in the domain specified in the wildcard certificate. For example, if the domain is ".fiware.org" a valid name may be "paasmanager.fiware.org".
- The clients should use this hostname instead of the IP
- The file /etc/keystorejetty must be replaced with another one generated from the wildcard certificate, the corresponding private key and other certificates signing the wild certificate.

It's possible that you already have a wild certificate securing your portal, but Apache server uses a different file format. A tool is provided to import a wildcard certificate, a private key and a chain of certificates, into /etc/keystorejetty:

```
# usually, on an Apache installation, the certificate files are at /etc/ssl/private
/opt/fiware-paas/bin/importcert.sh key.pem cert.crt chain.crt
```

If you have a different configuration, for example your organization has got its own PKI, please refer to: http://docs.codehaus.org/display/JETTY/How%2bto%2bconfigure%2bSSL

Configuring the PaaS Manager in the keystone

The FIWARE keystone is a endpoint catalogue which collects all the endpoint of the different services

1.3.5 Sanity check procedures

Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

End to End testing

Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. It involves to obtain the product information storaged in the catalogue. With it, we test that the service is running and the database configure correctly.

```
http://{PaaSManagerIP}:{port}/paasmanager/rest
```

The request to test it in the testbed should be

```
    curl -v -k -H 'Access-Control-Request-Method: GET' -H 'Content-Type: application xml'
        -H 'Accept: application/xml' -H 'X-Auth-Token: 5d035c3a29be41e0b7007383bdbbec57'
        -H 'Tenant-Id: 60b4125450fc4a109f50357894ba2e28'
        -X GET 'https://{PaaSManagerIP}:8443/paasmanager/rest/catalog/org/FIWARE/environment'
```

the option -k should be included in the case you have not changed the security configuration of PaaS Manager.

Whose result is the PaaS Manager API documentation.

List of Running Processes

Due to the PaaS Manager basically is running over the Tomcat, the list of processes must be only the Jetty and PostgreSQL. If we execute the following command:

```
ps -ewF | grep 'postgres\|jetty' | grep -v grep
```

It should show something similar to the following:

```
postgres  1327     1  0 58141  9256   0 08:26 ?        00:00:00 /usr/bin/postgres -D /var/lib/pgsql/d
postgres  1328  1327  0 48078  1696   0 08:26 ?        00:00:00 postgres: logger process
postgres  1330  1327  0 58166  3980   0 08:26 ?        00:00:00 postgres: checkpointer process
postgres  1331  1327  0 58141  2068   0 08:26 ?        00:00:00 postgres: writer process
postgres  1332  1327  0 58141  1808   0 08:26 ?        00:00:00 postgres: wal writer process
postgres  1333  1327  0 58349  3172   0 08:26 ?        00:00:00 postgres: autovacuum launcher process
postgres  1334  1327  0 48110  2052   0 08:26 ?        00:00:00 postgres: stats collector process
root     14054     1  4 598402 811464 0 09:35 ?        00:00:22 java -Xmx1024m -Xms1024m -Djetty.stat
postgres 14114  1327  0 58414  3956   0 09:36 ?        00:00:00 postgres: postgres paasmanager 127.0.
postgres 14117  1327  0 58449  3772   0 09:36 ?        00:00:00 postgres: postgres paasmanager 127.0.
postgres 14118  1327  0 58449  3776   0 09:36 ?        00:00:00 postgres: postgres paasmanager 127.0.
```

Network interfaces Up & Open

Taking into account the results of the ps commands in the previous section, we take the PID in order to know the information about the network interfaces up & open. To check the ports in use and listening, execute the command:

```
netstat -p -a | grep $PID
```

Where $PID is the PID of Java process obtained at the ps command described before, in the previous case 14054 jetty and 1327 (postgresql). The expected results for the postgres process must be something like this output:

```
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address         State       PID/Program name
tcp6       0      0 [::]:pcsync-https       [::]:*                  LISTEN      14054/java
tcp6       0      0 localhost:48017         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48015         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48027         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48016         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48022         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48023         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48029         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48013         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48012         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48019         localhost:postgres      ESTABLISHED 14054/java
```

```
tcp6       0      0 localhost:48028         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48014         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48020         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48024         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48031         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48021         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48018         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48026         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48030         localhost:postgres      ESTABLISHED 14054/java
tcp6       0      0 localhost:48025         localhost:postgres      ESTABLISHED 14054/java
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         STREAM     CONNECTED     71542    14054/java
unix  3      [ ]         STREAM     CONNECTED     71480    14054/java
```

and the following output for the jetty process:

```
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address         State       PID/Program name
tcp        0      0 localhost:postgres      0.0.0.0:*               LISTEN      1327/postgres
tcp6       0      0 localhost:postgres      [::]:*                  LISTEN      1327/postgres
udp6       0      0 localhost:53966         localhost:53966         ESTABLISHED 1327/postgres
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     19508    1327/postgres        /tmp/.s.PGSQL.5432
unix  2      [ ACC ]     STREAM     LISTENING     19506    1327/postgres        /var/run/postgresql/
```

Databases

The last step in the sanity check, once that we have identified the processes and ports is to check the different databases that have to be up and accept queries. Fort he first one, if we execute the following commands:

```
psql -U postgres -d paasmanager
```

For obtaining the tables in the database, just use

```
paasmanager=# \dt

 Schema |              Name                       | Type  |  Owner
--------+-----------------------------------------+-------+----------
public  | applicationinstance                     | tabla | postgres
public  | applicationrelease                      | tabla | postgres
public  | applicationrelease_applicationrelease   | tabla | postgres
public  | applicationrelease_artifact             | tabla | postgres
...
```

1.3.6 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

Resource availability

The resource availability should be at least 1Gb of RAM and 6GB of Hard disk in order to prevent enabler's bad performance. This means that bellow these thresholds the enabler is likely to experience problems or bad performance.

Resource consumption

State the amount of resources that are abnormally high or low. This applies to RAM, CPU and I/O. For this purpose we have differentiated between:

- Low usage, in which we check the resources that the Tomcat requires in order to load the PaaS Manager.
- High usage, in which we send 100 concurrent accesses to the PaaS Manager.

The results were obtained with a top command execution over the following machine configuration:

Name	Type
Type Machine	Virtual Machine
CPU	1 core @ 2,4Ghz
RAM	1,4GB
HDD	9,25GB
Operating System	CentOS 6.3

The results of requirements both RAM, CPU and I/O to HDD is shown in the following table:

Resource Consumption	Low UsageType	High Usage
RAM	1GB ~ 63%	3GB ~ 78%
CPU	0,8% of a 2400MHz	90% of a 2400MHZ
I/O HDD	6GB	6GB